**Difference between JPA, Hibernate and Spring Data JPA**

**Java Persistence API (JPA)**

* **JPA** is a **Java Specification (JSR 338)** for mapping Java objects to relational databases.
* It defines a **standard** for:
  + Creating, reading, updating, deleting Java objects from DB (CRUD).
  + Querying using JPQL.
* **Important**: JPA is **only an API** – it does **not provide an actual implementation**.

**Hibernate**

* **Hibernate** is a **popular ORM (Object-Relational Mapping) tool**.
* It provides a **concrete implementation of JPA**.
* You can use Hibernate **with or without JPA annotations**.

Hibernate gives:

* Caching
* Lazy loading
* Dirty checking
* HQL (Hibernate Query Language)

**Spring Data JPA**

**Spring Data JPA** is a **Spring-based abstraction** over JPA.

It:

* **Does NOT implement JPA**
* Uses a provider like **Hibernate** internally
* Reduces **boilerplate code** (no need to write Session, Transaction, etc.)
* Manages transactions automatically
* Helps with powerful query methods like findByNameAndAge()

**Comparison: Hibernate vs Spring Data JPA**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Hibernate (Traditional)** | **Spring Data JPA** |
| Boilerplate Code | More | Much less |
| Session Management | Manual | Auto-managed |
| Transaction Management | Manual | Auto via @Transactional |
| Repositories | No built-in support | Uses JpaRepository |
| Queries | HQL / Criteria | Derived queries / JPQL / Native |

**Code Comparison**

Hibernate Code

public Integer addEmployee(Employee employee){

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

return employeeID;

}

Spring Data JPA Code

**EmployeeRepository.java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

EmployeeService.java

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}